

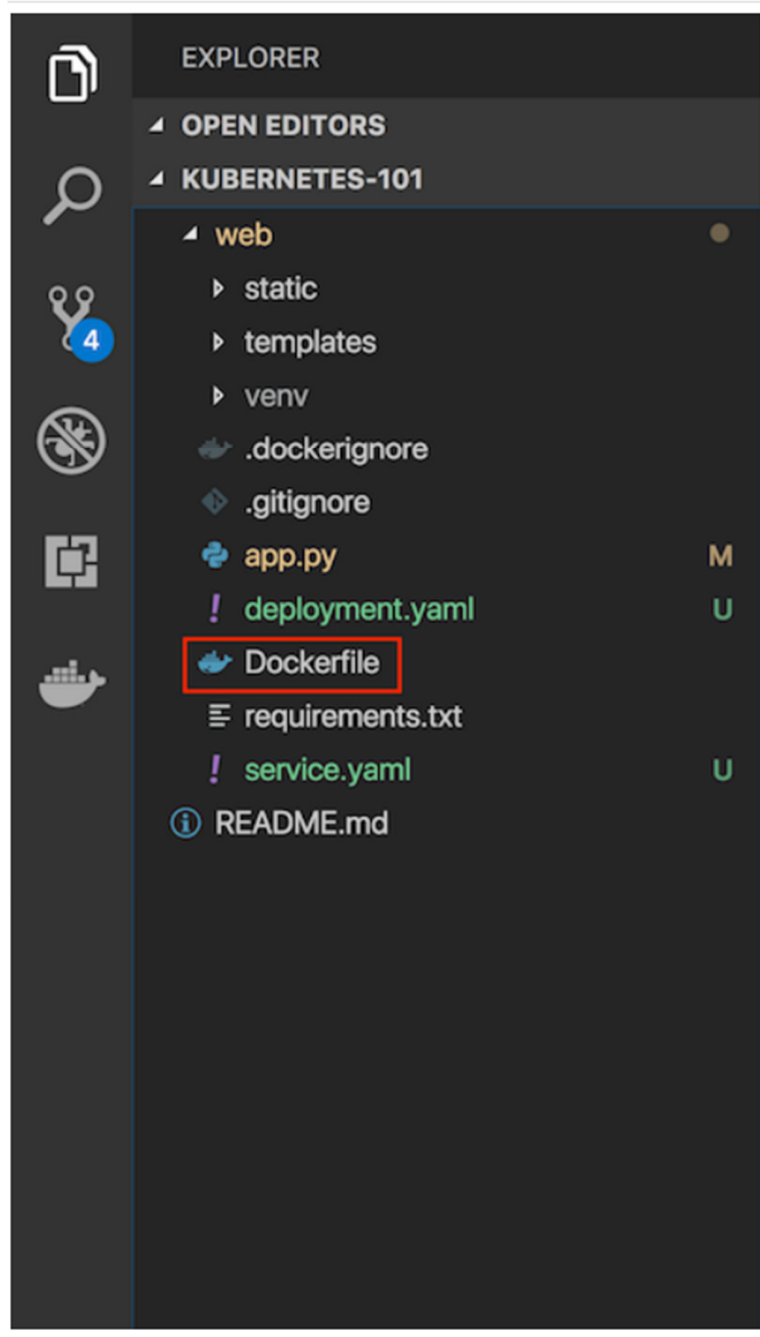
DEPLOYMENT OF APP IN IBM CLOUD

CONTAINERIZE YOUR FLASK APPLICATION

TEAM ID	PNT2022TMID27679
PROJECT NAME	SKILL/JOB RECOMMENDER APPLICATION

Containerize your Flask application

- In your project directory, create a file named "Docker file." *Suggestion: Name your file exactly "Docker file," nothing else.*



A "Docker file" is used to indicate to Docker a base image, the Docker settings you need, and a list of commands you would like to have executed to prepare and start your new container.

In the file, paste this code:

```
FROM python:2.7
LABEL maintainer="Kunal Malhotra, kunal.malhotra1@ibm.com"
RUN apt-get update
RUN mkdir /app
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
EXPOSE 5000
ENTRYPOINT [ "python" ]
CMD [ "app.py" ]
```

Explanation and breakdown of the above Docker file code

1.The first part of the code above is:

```
FROM python:2.7
```

Because this Flask application uses Python 2.7, we want an environment that supports it and already has it installed. Fortunately, DockerHub has an official image that's installed on top of Ubuntu. In one line, we will have a base Ubuntu image with Python 2.7, virtualenv, and pip. There are tons of images on DockerHub, but if you would like to start off with a fresh Ubuntu image and build on top of it, you could do that.

2.Let's look at the next part of the code:

```
LABEL maintainer="Kunal Malhotra, kunal.malhotra1@ibm.com"
RUN apt-get update
```

3.Note the maintainer and update the Ubuntu package index. The command is RUN, which is a function that runs the command after it.

```
RUN mkdir /app  
WORKDIR /app  
COPY . /app
```

4. Now it's time to add the Flask application to the image. For simplicity, copy the application under the /app directory on our Docker Image.

WORKDIR is essentially a **cd** in bash, and COPY copies a certain directory to the provided directory in an image. ADD is another command that does the same thing as COPY, but it also allows you to add a repository from a URL. Thus, if you want to clone your git repository instead of copying it from your local repository (for staging and production purposes), you can use that. COPY, however, should be used most of the time unless you have a URL.

5. Now that we have our repository copied to the image, we will install all of our dependencies, which is defined in the requirements.txt part of the code.

```
RUN pip install --no-cache-dir -r requirements.txt
```

6. We want to expose the port(5000) the Flask application runs on, so we use EXPOSE.

```
EXPOSE 5000
```

7. ENTRYPOINT specifies the entry point of your application.

```
ENTRYPOINT [ "python" ]  
CMD [ "app.py" ]
```

Build an image from the Docker file

Open the terminal and type this command to build an image from your Docker file: `docker build -t <image_name>:<tag>` . (note the period to indicate we're in our apps top level directory). For example: `docker build -t app: latest` .

```
kunals-rbp:web kunalmlhotra$ docker build -t app:latest .
Sending build context to Docker daemon 348.2kB
Step 1/8 : FROM python:2.7
--> 6c76c9a7cfe
Step 2/8 : LABEL maintainer="Kunal Malhotra, kunal.malhotra@libn.com"
--> Using cache
--> d8957d41531c
Step 3/8 : RUN apt-get update
--> Using cache
--> 6062a134e40e
Step 4/8 : COPY . /app
--> f0d7f27a931f
Step 5/8 : WORKDIR /app
Removing intermediate container f9010699a2fe
--> 0bcc6af20e3d
Step 6/8 : RUN pip install -r requirements.txt
--> Running in 8153040b0007
Collecting click==6.7 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/34/c1/8886f99713dd993c5366c362b7f988f18269f8d793af1fab4700775a77/click-6.7-py3-none-any.whl (71kB)
Collecting Flask==1.0.2 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/7f/47/80576774ed4536d324b14d0cb469638663460af824ea997202c08ed4b/Flask-1.0.2-py2.py3-none-any.whl (93kB)
Collecting itsdangerous==0.24 (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/dc/b4/060bcb945c00f6d088d8375131ab3f25b22f2bfcfd0b221165194b204/itsdangerous-0.24.tar.gz (40kB)
Collecting Jinja2==2.10 (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/7f/7f/0e6f0c4cf35f27a016a7bed8e86730e4d277a78ca76f32659220a731/Jinja2-2.10-py2.py3-none-any.whl (126kB)
Collecting MarkupSafe==1.0 (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/4d/de/530471db3160f4b76808226d57001ef7a448255de9693ab4bfcd4f4172b/MarkupSafe-1.0.tar.gz
Collecting Werkzeug==0.14.1 (from -r requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/20/c4/72e3e56473e5375aa79c4764e70d1b83ef0687bef8d0a0e4fe335243/Werkzeug-0.14.1-py2.py3-none-any.whl (320kB)
Building wheels for collected packages: itsdangerous, MarkupSafe
  Running setup.py bdist_wheel for itsdangerous: started
  Running setup.py bdist_wheel for itsdangerous: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/2c/4a/61/5599631c1554768c620b08d82c72d7317910374ca682ff1e5
  Running setup.py bdist_wheel for MarkupSafe: started
  Running setup.py bdist_wheel for MarkupSafe: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/25/26/20/6ee40ac0127f1e1c0832340b1699f969467676081e4e4e
Successfully built itsdangerous MarkupSafe
Installing collected packages: click, itsdangerous, MarkupSafe, Jinja2, Werkzeug, Flask
Successfully installed Flask-1.0.2 Jinja2-2.10 MarkupSafe-1.0 Werkzeug-0.14.1 click-6.7 itsdangerous-0.24
Removing intermediate container 8153040b0007
--> 46d6336a270c
Step 7/8 : ENTRYPOINT [ "python" ]
--> Running in bd1c83815e1
Removing intermediate container bd1c83815e1
--> 73cef380c1c
Step 8/8 : CMD [ "app.py" ]
--> Running in a7849430a46f
Removing intermediate container a7849430a46f
--> d806a83763e5
Successfully built d806a83763e5
Successfully tagged app:latest
kunals-rbp:web kunalmlhotra$
```

Run your container locally and test

After you build your image successfully, type: `docker run -d -p 5000:5000 app`

This command will create a container that contains all the application code and dependencies from the image and runs it locally.

```
kunals-rbp:web kunalmlhotra$ docker run -d -p 5000:5000 app
3c2ba786f758a0a80a6e52ceef380ea840a8263137ca5543c0bcb16247
kunals-rbp:web kunalmlhotra$ docker ps

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
3c2ba786f758	app	"python app.py"	Less than a second ago	Up 5 seconds	0.0.0.0:5000->5000/tcp	compassionate_keldysh

```
kunals-rbp:web kunalmlhotra$
```

